UNITED STATES

PATENT APPLICATION

FOR

IMAGE MANIPULATION METHOD AND SYSTEM

Inventors:

Michael Ferraro
Michael Sweet

Express Mail Label No.: EJ146546736US

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 USC 119(e)(1) of provisional patent application Serial No. 60/165,822 filed November 16, 1999.

COPYRIGHT RIGHTS

BACKGROUND OF THE INVENTION

Animation in its various incarnations can be extremely labor intensive. In its most traditional form, individual cels must be drawn so that a sequencing through the cels in quick succession produces the desired effect. With this technique, if an individual cel is later found to be incorrect or a change in the animation is desired, an entire group of cels may need to be discarded.

The advent of the computer has enabled the production of individual cels to be sped up to some extent and facilitated easier editing of existing cels. However, conventional computer assisted animation techniques still do not readily allow changes to the animation to be made "on the fly" or in the middle of a sequence without the need to manually re-edit several, if not many, frames thereafter. Moreover, because most animation, is based upon a sequence of key frames, a single change affecting timing can skew the rest of the animation, such that its synchronization with audio is jeopardized, if not ruined. This is a particular problem when the synchronization

relates to the correspondence of mouth images with the audio. A skew between the audio and images can result in a finished product that looks like a poorly dubbed movie. Additionally, if an animation is created for one language, it is currently difficult, if not impossible, to reproduce the same animation in another language in a quick and cost-efficient manner.

Additionally, once an animation is completed the individual cels are typically unusable to produce a new animation, unless the cels are used as is. Thus, in order to produce a series of episodes of a show including animations, new animation cels must be created for each episode, thereby significantly increasing the cost of creating the series.

Moreover, the process of creating animation sequences using traditional techniques also makes it difficult, if not impossible, to have an animated character interact with a living person in a live and spontaneous or impromptu manner without any prior coordination between them.

Furthermore, there is presently no cost effective, non-highly labor intensive way to re-use or generate new footage from old cells or film footage.

SUMMARY OF THE INVENTION

Thus there is a need in the art for an arrangement that does not suffer from the drawbacks of the prior art. Advantageously, applying the teachings herein allows for creation of sophisticated animations in a cost-efficient and less labor intensive manner than often present in the prior art.

In the interest of brevity, an understanding of animation systems and techniques, such as described in Kit Laybourne, "The Animation Book: A Complete Guide to Animated Filmmaking-From Flip-Books to Sound Cartoons to 3-D Animation" is presumed. Moreover, it is to be understood that, by applying the principles of the invention, aspects of the animation

techniques described in Laybourne may be advantageously employed, augmented, replaced or surpassed.

In general, in a first aspect, the invention features an animation system made up of an audio analysis unit having an audio output and a state change output and a control unit having an input and an output, the input being constructed to receive a state change indication from the audio analysis unit, the control unit being further constructed to generate compositing command information for individual component elements of an animation frame and send the compositing command information out the output.

In general, in a second aspect, the invention involves a method of creating an animation from a set of actions. The method specifically involves receiving a set of signals reflecting the set of actions, outputting a set of compositing commands based upon the set of signals, the set of compositing commands being related to the set of signals by sequences of transition data items; and compositing first images into a first animation using the set of compositing commands and a first graphic database.

Still other aspects involve numerous additional features and provide further advantages beyond those set forth herein, the enumerated advantages and features described herein being a few of the many advantages and features available from representative embodiments. These enumerated advantages and/or features are presented only to assist in understanding the invention. It should be understood that they are not to be considered limitations on the invention as defined by the claims, or limitations on equivalents to the claims. For instance, some of these advantages are mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some advantages are applicable to one aspect of the invention, and inapplicable to others. Thus, the specifically referred to features and advantages should not be

considered dispositive in determining equivalence. Additional features and advantages of the invention will become apparent in the following description, from the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1A is a high level functional block diagram representation of an example system constructed to apply the principles of the invention;

Fig. 1B is a high level functional block diagram of an alternate example system constructed to apply the principles of the invention;

Figs. 2A-2D illustrate possible physical configurations or distributions of functional units within computer hardware;

Fig. 3 is one example embodiment of the system logically shown in Figs. 1 and 2B;

Fig. 4 is a functional block diagram of an example Audio Analysis Unit in accordance with the invention;

Figs. 5A-5B are example arrangements of components of the Audio Analysis Unit in accordance with the invention;

Fig. 6, is a representative audio input signal;

Fig. 7 is a sample output resulting from performing a Fast Fourier Transform (FFT) on an audio input signal;

Fig. 8 is a functional block diagram of an Signal Analysis Module according to the principles of the invention;

Figs. 9A and 9B are each functional block diagrams of Mouth Classification Unit variants according to the principles of the invention;

Figs. 10A-10R and 10S-10LL are two examples of software implementing aspects of the Audio Analysis Unit according to the principles of the invention;

Fig. 11 is an example of the basic components of a character and how they are arranged in relative priority or layer order;

Fig. 12 shows four example poses that the character of Fig. 11 can assume;

Fig. 13 is an example gesture table for the character of Fig. 11 in Pose A;

Fig. 14A illustrates selected images from a frames database for an in-pose transition of the character of Fig. 11;

Fig. 14B illustrates selected images from a frames database for a pose to pose transition of the character of Fig. 11;

Fig. 15 is an illustration of a portion of a transition table for the character of Fig. 11;

Fig. 16 is an illustration of entries that can be stored in transition table cells;

Figs. 17 through 75 are a program listing of one example embodiment of a Compositing Engine suitable for use according to the principles of the invention;

Figs. 76 through 78 are an example of a software embodiment of an Audio Analysis Unit according to the principles of the invention;

Figs. 79 through 88 roughly implement an example software embodiment of a Transition Control Unit for the different components of the character of Figs. 11 through 15; and

Fig. 89 is an example procedural for the character of Fig. 11.

DETAILED DESCRIPTION

The principles of the invention and certain representative applications of those principles in various embodiments will be evident from the following discussion with reference to the Figs..

Fig. 1A is a high level functional block diagram representation of an example system constructed to apply the principles of the invention. The system includes, as primary functional components, an audio analysis unit (AAU) 100, a Cueing Unit 110, a Transition Control Unit (TCU) 120, and a Compositing Engine (CE) 130.

Audio Analysis Unit (AAU)

The AAU 100 is made up of an audio delay 102, a syncronization signal generator 104, and an signal analysis unit 106 (SAU). The AAU 100 has an input circuit 108 connected to the audio delay 102 and the signal analysis unit 106. The audio delay 102 delays an input audio signal for the amount of time necessary to allow processing to occur and be synchronized with the corresponding generated images. Alternatively, or additionally, in some cases the input audio signal can be advanced so that the combination of the advance and delay results in the proper overall delay for resynchronization. The amount of delay is adjustable or programmable to flexibly allow, for example, usage of the system at film frame speeds of 64, 24, 18 or 8 frames per second, video frame speeds of 30 frames per second, computer frame speeds of 15 or 8 frames per second, or other desired standard or non-standard frame speeds while maintaining syncrony between audio and video. The syncronization generator 104 is a "clock" applied to various system components and is specifically used to maintain a consistent frame rate among the system components involved in frame generation. It is the combination of the audio delay 102 and syncronization generator 104 which maintains the syncronization of the visual

components of a frame with the audio input signal, for example, to produce animation that has a character's speech syncronized with the visual animation of the character's mouth, commonly referred to as "lip-sync."

To accomplish the lip-sync, the signal analysis unit 106 uses digital signal processing techniques to perform an analysis of the input audio signal and outputs information which is used to associate the input audio signal stream with a series of mouth images. The output of the signal analysis unit 106 is a series of individual data signals each representing a "mouth state." In one embodiment, the signal analysis unit 106 performs a Fast Fourier Transform (FFT) on the audio input signal and analyzes the result to make associations between the characteristic of a sound and the mouth shape that is used to articulate that sound.

In general, the specific correlation between spoken sounds and mouth shapes depends upon the preferences of each particular animator and may even vary among characters animated by the same animator. In one embodiment, one feature of the signal analysis unit 106 is that it is tunable. This allows different animators to use the same system while accounting for their own preferences in spoken sound/mouth correlation as well as for processing and syncronization of different tones of voice, individual character traits or speaker dialects, and/or non-phonetic languages. In other embodiments, the tuned settings may be stored and retrieved so that two or more different animators can share the same system for different projects and time is not wasted re-tuning for each animator's preferred settings. As will also become apparent from the detailed description below, alternative embodiments of the signal analysis unit 106 can also be used to syncronize non-vocal input audio signals with character movements and/or other changes, for example, to allow an animated character to dance based upon input music or react in some

controlled way to particular sounds such as a honking horn, a gunshot or the buzzing of an insect.

Cueing Unit

The cueing unit 110 is functionally a control unit made up of a cue sheet 112 and a cue controller 114. The cue sheet 112 is a logical construct which describes a point in time where a particular action, or change in the animation frame, is supposed to take place, for example, relative to some specified reference. By way of a particular example, the cue sheet 112 can specify that at time t_0, use background 1; at time t_5 (or k milliseconds from the start of the audio), switch to background 2; at time t_{100}, introduce a new character, etc. The cue sheet 112 is an input to the cue controller 114.

The cue controller 114 clocks time, for example, relative to the start of the audio; and compares it against the cue sheet 112. When the cue controller 114 finds a match between an entry in the cue sheet 112 and the current elapsed time for the audio signal, it sends out data representing a change of state. As will be described in greater detail below, this state change information is used to signal or effect a change in the animation, for example, to cause a background change or initiate an action sequence.

As shown, the cue controller 114 optionally includes an input for a Society of Motion Picture and Television Engineers (SMPTE) time code which is a time code that is commonly written onto video tapes and can be used as a standard reference. This allows, as an added convenience, the cue sheet 112 to use SMPTE times as an alternate reference. In this manner, if an audio track or videotape has been pre-recorded with SMPTE times, and key events are to occur at prescribed times, the cues can be indicated as needing to match a particular SMPTE

time as opposed to an elapsed audio time. Of course, the cue sheet 112 would then hold SMPTE time codes instead of elapsed audio time.

Transition Control Unit (TCU)

The TCU 120 is functionally made up of one or more controllers, illustratively shown as a discrete trigger controller 122, a continuous controller 124, a programmed event controller 126 and a transition controller 128. These controllers are the primary control mechanism. Using one or more of these controllers, an animation can be "performed" in a manner somewhat analogous to a puppeteer controlling a puppet except, as will become apparent from the discussion below and unlike a puppet, any component of an animation frame can be controlled.

The discrete trigger controller 122 and continuous controller 124 manage triggers coming in from input devices, for example, triggers generated by MIDI boards, serial control devices, contact closures, analog to digital converters, etc . . . In general, discrete trigger controllers 122 and continuous controllers 124 are similar in function to the cue controller 114, in that, when a trigger is applied to one of the controllers 122, 124 and it matches a specified trigger action in that controller 122, 124, an event is generated or "fired off" to cause a state change to take place. The state change can include, for example, a change in head position, eye position, or mouth position; a costume change, a translation or rotation in a plane or in space, a camera change, or a compound change (e.g. head position and face color, or arm position, face color and mouth position), etc. In short, virtually any state change can be set up to be fired off with a trigger.

Different discrete trigger controllers 122 generally handle different kinds of trigger actions. For example, an instantaneous trigger, like a button or piano keyboard key, can fire off one trigger when the button or key goes both down and up. Alternatively, the button can be used to produce separate state changes for the down transition and the up transition.

A continuous controller 124 handles trigger actions which generate a continuous or time varying state change rather than an instantaneous change. A few examples of the many trigger actions that are appropriate for a continuous controller 124 are: movement of a wheel or slider on an audio mixing console, the motion of a computer joystick, depressing a pedal such as is used by musicians with keyboards or guitars, or motion along a ribbon on a MIDI keyboard which registers a continuous change as a finger is slid along it for as long as the finger is in motion, an ergonomic, goniometric, kinetic and/or telemetric input device for identifying, responding to, or controlling movement, a motion tracking or capture suit, etc.

A programmed event controller 126 is a controller which fires off pre-specified events as opposed to requiring external trigger actions. Examples of programmed trigger actions include firing off events at some random interval to cause eye blinks or lightning flashes, causing trigger events based upon the time of day or according to some specified algorithm in order to cause a state change.

Depending upon the particular implementation, a system may have none, one, or more than one of each type of controller 122, 124, 126. Additionally, two or more different types of controllers can be part of a single device.

State changes indicated by the controllers 122, 124, 126, as well as state changes and mouth states respectively output by the cue controller 114 and audio analysis module 100 are applied as inputs to the transition controller 128. The transition controller 128 uses the input state changes and/or mouth state and creates a goal state, which is the desired state to which the animation frame will be brought. It also uses transition tables 130 to resolve the difference between the current state and the goal state. The transition controller 130 cycles through a series of actions in order to bring the current state of frame elements in concordance with the goal state

of frame elements on a frame-by-frame basis. Feedback 132 is used to allow the transition controller 128 to know when the goal state has been reached. When a trigger signals that an action is required to bring the current state toward the goal state, the transition controller 128 outputs commands which cause a component of the frame or element of the character to be updated, in response to the trigger signals, to reflect some kind of motion or change. The compositing commands consist of, for example, commands that cause hiding and showing of elements; moving an element around on the screen; changing the color or order in which the elements are drawn on the screen; or others which affect the appearance of the character or look at any given frame.

Although described in greater detail below, in overview, transition tables 131 define the allowed sequential steps between initial states and goal states. For example, a particular trigger may indicate a movement of a character's arm from its hip down to its side. A series of steps would be used to get the arm from the initial state (onHip) to another state (atSide). An entry in the transition table 131 would include the particular series of sequential steps that would be required to effect that change in state which would ultimately be reflected in the images of sequential animation frames.

Another output coming out of the transition controller 128 is the current state. This current state is fed back into the transition controller 128 in order to monitor for when the goal state is achieved. As the transition controller 128 optionally brings the character to the goal state, all actions that are required to achieve that result are optionally recorded in an optional time line recorder 134. This record is useful, for example, for quickly and efficiently redubbing in a new language, changing character looks, etc. It can also be used to "sweeten" a performance after it has been captured and then re-layed down to tape. Additionally the time line recorder

134 allows a real time performance to be captured and, if necessary, converted to other animation file formats using a Format Convertor 136, such as Macromedia Flash. In this manner, an animation can be streamed in a format compatible with current popular internet browsers such as Netscape Navigator or Internet Explorer over the internet. Additionally, by opening a socket to the browser, an animation can be performed live. Since the process can occur in real time (as used herein, "real time" means the time (Δt) between the input of an audio signal or trigger at a time (t_x) and the compositing of a frame incorporating the effect of the input at t_x will be less than the frame speed (i.e., $\Delta t <$ frame speed)), animation for web casting can be produced very quickly, much quicker than can be done, for example, in Flash by itself.

Compositing Engine (CE)

The CE 130 is the unit that creates the viewable portion of the animation based upon the output of the transition controller 128. In one embodiment, the CE 130 creates 2-dimensional animation frames in a conventional manner. In another alternative embodiment, the compositing engine 130 is designed to create 3-dimensional animation frames. This embodiment of the compositing engine is built around a notion of taking a transparent piece of geometry, called an Idolon (after an Eidolon or phantom), and on to that piece of geometry is hung an image which can be positioned in space, rotated, scaled and otherwise contorted in real-time, the transparency can also be adjusted in real-time with the image or independently. In the case where the Idolon and image move as one, the combination is referred to as a Scrim. A frame database 132 is one of the inputs into the compositing engine 130. The frame database 132 is a database of images of all the components of the character in all possible usable states, as well as other visual elements of a frame subject to change. The TCU 120 and the transition tables 131 sequence and layer those elements in the proper order so that when the images are composited using the CE

130, the desired effect is achieved. Compositing is conventionally done using software or hardware. In fact, graphics hardware often has a built in compositing program which is used for matting out parts of an image while permitting other parts of the image to show through. Alternatively, a custom compositing engine can be written in software, such as shown in Figs. 17 through 75. The output of the CE 130 is a stream of graphic elements for the frames which are sent to a frame buffer 138. The frame buffer 138 is generally also built into the video hardware or accelerating display hardware that is used to produce the video output and may be of any suitable type commercially available. Some representative examples of suitable display hardware are the DigiMotion, DigiMix and DigiSuite from Matrox Electronic Systems Ltd., 1055 St. Regis Blvd, Dorval, Quebec, Canada, H9P 2T4, and the Tornado 3000 from Evans & Sutherland, 600 Komas Drive, Salt Lake City, UT 84108.

Fig. 1B is a high level functional block diagram of an alternate example system constructed to apply the principles of the invention. The system is similar to the system of Fig. 1A except that the system of Fig. 1B includes a Programmed Event Controller and Controller Pass-thru (PECP) Unit 127. The PECP Unit 127 is contructed to receive state change indications from other controllers and, in some cases, pass them directly through to the Transition Controller 128 and, in other cases, trap the state change indications and use them as an internal trigger for a programmed event sequence or, in still other cases, both pass the state change indications and use them internally. In the example system of Fig. 1B, the PECP Unit 127 can receive and trap and/or pass state change indications from the Signal Analysis Unit 106, Cue Controller 114, Trigger Controller 122 and the Continuous Controller 124. Additionally, in some variants, the PECP Unit 127 can be capable of receiving SMPTE input directly for internal use (for example, as a key for a programmed event sequence). In operation, the programmed event controller

functionality of the PECP Unit 127 is similar to the Programmed Event Controller 126 of Fig. 1A.

An optional Outline (Vector) Database 133 is also included in the system of Fig. 1B. The Outline Database 133 is similar in function to the Frames Database 132 except instead of containing actual images of components, for example, in bitmap form, the Outline Database 133 contains drawing outline or "vector" information that can be manipulated and/or filled in by separate algorithms running in the CE 130. The contents of the Outline Database 133 can be understood, by contrasting it, with the Frames Database 132. In a simple example, a two dimensional representation of a cube would be stored in the frames database as a number of discrete images taken from particular angles of view. Every visible face of the cube that had a color would be colored as well. In contrast, in the Outline Database 133, the same cube would be represented as data that would produce a "wire frame" or outline image of the cube. Hidden line and polygon fill algorithms running in the CE 130 would then be used to place the cube in the desired perspective position and fill the sides of the cube appropriately.

Alternatively, a combination of the two databases 132, 133 could be used, with the Outline Database 133 containing the outline information for a character component or shape and the Frames Database 132 containing image information to be placed within the outline.

It should also be understood that, while the optional Outline Database 133 has been shown and described in connection with Fig. 1B, it could be readily employed in, for example, the system of Fig. 1A or virtually any other variant thereof described herein.

Referring now to Figs. 2A-2D, there are at least several possible physical configurations or distributions of these functional units within computer hardware. In the first example (Fig. 2A), the AAU 100, TCU 120 and CE 130 are all contained in one computer 200. In the second

example (Fig. 2B), the AAU 100 is logically in one computer 202, and the TCU 120 and CE 130 are both logically in a second computer 204. In the third example (Fig. 2C), the AAU 100 and the TCU 120 are both logically in the same computer 206, whereas the CE 130 is logically on its own computer 208. In the fourth example (Fig. 2D), each of the AAU 100, TCU 120, and CE 130 are logically on their own computers 210, 212, 214. It will also be apparent that any logical unit may be distributed among two or more physical computers, that portions of two or more logical units can be in the same physical computer, and that any of the individual computers can be physically made up of two or more independent processors in a parallel processing, loosely coupled, or tightly coupled arrangement if more speed or processing power is desired for a particular logical unit.

In the simplest embodiment, the logical units described above, the AAU 100, the TCU 120, and the CE 130 are all physically part of a single computer. One such suitable example computer is a 300 MHz Macintosh PowerBook. This embodiment has the advantage of portability and can, for example, be advantageously employed as a demonstration unit for rehearsing an animation, as a mini production studio, or to provide low cost, on-site or remote editing capability, etc.

In another embodiment involving producing material for a television show where a lot of elements are moving on the screen at one time, the AAU 100 can, for example, be separated from the TCU 120 CE 130. One example impetus for doing so, is that the demands of a television show with multiple simultaneous actions can require a great deal of computer power, particularly if the usage requires the animation to run within a real-time window at a typical rate of 15 frames per second (66 milliseconds). The AAU 100 is therefore embodied in a separate

computer 202 to allow the TCU 120 and CE 130 to consume most of the CPU power of their host computer 204 for the animation output.

In still other embodiments, where even more computer power is required for compositing, the CE 130 can be embodied on its own computer 214, with or without special purpose hardware. One example embodiment of this type employs a high speed, powerful computer 214 for dedicated use as compositing engine 130. Suitable example computers for this application are the Silicon Graphics (SGI) Model O2 from Silicon Graphic Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, or the Z series computers from Intergraph Corp., Huntsville, AL 35894-0001, the latter also specifically including hardware suitable for use in conjunction with the TCU 120 and/or CE 130. Built-in graphics acceleration and, for example, OpenGL software can also be employed to composite in real-time using 3-D acceleration hardware and frame buffer acceleration. Where multiple computers are employed (Figs. 2C-2D), information transferred among the various units is done according to known techniques and protocols appropriate for the situation including, for example, MIDI, Ethernet, and optical fiber, to name a few. Moreover, by employing known techniques such as multiplexing, several AAUs can share a common TCU, several TCUs can share a common CE, and/or a multiplexed output of a TCU can be demultiplexed for compositing by two or more CEs. In the case where multiple CEs are used, they may share a common Frames Database 132 and/or Outline Database 133 as desired or each employ their own discrete databases. As to the individual computers themselves, other suitable computers are any commercially available computers incorporating the Unified Memory Architecture (UMA) or an analogous memory usage scheme, such as employed, for example, in the CyberBlade i7 chip from Trident

Microsystems, Inc., 2450 Walsh Avenue, Santa Clara, CA 95051 or the Apollo MVP4 chip set from Via Technologies, Inc., 1045 Mission Court, Fremont, CA 94539.

Fig. 3 is one example embodiment of the system logically shown in Figs. 1 and 2B. In this embodiment, the AAU 100 runs on a Macintosh G-3 300, 300 MHz or higher, having approximately 256 megabytes of RAM or more, and having disk storage of sufficient size to hold digitized audio as appropriate. As shown, the disk storage is a single 4Gbyte hard disk drive, however other suitable storages such as solid state disks and/or disk arrays my alternatively be used. A multi-channel analog and digital audio interface unit 302 connects to the Macintosh 300 through a PCI card (not shown) via a high speed connection 304 between the interface unit 302 and the PCI card, for example, a high speed serial audio wire. There are several suitable commercially available audio interface units 302, one such example of a suitable audio interface unit 302 is model MOTU2408, commercially available from Mark of The Unicorn, 1280 Massachusetts Avenue, Cambridge, MA 02138, which has analog audio inputs, digital audio inputs, and multiple inputs and outputs for both analog and digital.

The purpose of the audio interface unit 302 is to convert audio input into a digital data stream. By way of some representative examples, a microphone connected to the analog audio input channel would be converted to a digital signal and then transmitted to the Macintosh G-3 300 as a stream of digital audio data. Similarly, the audio output of a video tape deck when input into one of the analog inputs, would also be presented to the Macintosh as a digital data stream. Another alternative would be to send a digital audio stream from a digital audio tape (DAT), a stored digitized file or from the output of a digital video deck such as Sony Digibeta (a digital Sony Beta recorder playback deck) into the digital inputs of the MOTU2048 302 which, in turn, presents it to the Macintosh 300 as a digital audio stream. The delayed audio from the AAU 100

also comes out of the audio interface unit 302 and is then suitable for displays in sync with the frame output, for example, on a video tape deck.

The AAU 100 in one embodiment is software which, when executing, examines the audio stream, converts it to an FFT, does some analysis, does some classification, and outputs a mouth state. The mouth state is basically a character identification associated with the channel on which the audio input stream is being presented, and which is appropriate for the given sound.

The mouth state is then presented to the TCU 120 shown, for purposed of illustration, as running in a second Macintosh 306. The connection 308 between the two computers 300, 306 is illustratively shown as standard MIDI or ethernet connections, although optical or other connections of suitable speed and bandwidth may also be used .

A series of conventional MIDI controllers 310-1, 310-2 , 310-K for example, those contained on the "Launch Pad" by E-MU/ENSONIQ, 600 Green Hills Road, P.O. Box 660015, Scotts Valley, CA 95067-0015 are connected to the TCU 120. The Launch Pad product has a series of octave twelve buttons that record an up and a down state, four sliders for continuous control, buttons that only record a down state, a knob that is spring loaded for continuous variation, pedal input, etc. Other alternative inputs could be any other general or special purpose MIDI-control device such as commonly used by musicians. The output is standard MIDI data, which is presented to the TCU 120 as a trigger occurring on a particular channel. Depending upon the implementation, multiple MIDI inputs can be fed into the computer hosting the TCU 120 at once. In others, they can go through a large MIDI interface supporting multiple inputs. In any event, the triggers are presented to the computer as stream of data associated with a channel. The trigger has a value associated with it which is processed by one of the trigger controllers 122, 124. A relationship between a channel and a

trigger is made such that, when a trigger is presented with a certain value, a state change is output and sent the TCU 120. It is with these devices that character movement, such as head turns, costume changes, background changes, etc., are produced. For example, button 1 on a MIDI control channel may cause the head of character 1 to go from whatever position it is currently in to a profile left position. The same button on controller 2 could be used to bring character 2's head to its profile position. It is noteworthy that the association between trigger and action is arbitrary and is actually made in the trigger controllers themselves which, as noted above, are functionally considered part of the TCU 120. In most instances, the trigger controllers 122, 124 are simple software elements that make the desired associations. Alternatively, one could use a patch bay.

On the output side of the TCU 120, an available option is to use a video card which converts the frame buffer pixels to an analog video signal. Alternatives cards can be employed if there is a need to which produce a digital video signal. One of the many acceptable video cards is called a "Targa 2000 Video Pro" made by Pinnacle Systems, 280 N. Bernardo Avenue, Mountain View, CA, 94043, which outputs an image that is displayable on the screen. The video card converts the digital video signal to a component video signal, which is a professional grade video signal suitable for input into broadcast quality video tape decks 312. In another embodiment, the output can be sent to a video switcher 314 for use in a television studio. In yet another embodiment, it can be sent to a large screen or projector. In short, it will be apparent that the output could readily be formatted and sent to any video display or recording device. For example, in the case of the Targa 2000, it permits the generation of not only professional grade component video, it can also output a consumer grade NTSC composite signal. In yet other

embodiments, the output of the frame buffer 138 can be formatted and stored for use by some other application or in some other manner.

Fig. 4 is a functional block diagram of the AAU 100. This functional unit can be viewed as being made up of two primary subcomponents. One of them is called the Signal Analysis Module 402 (SAM), and the other is the Mouth Control or Mouth Classification Unit 404 (MCU). The SAM 402 is where a signal analysis is performed and the MCU 404 is where the mouth classification is performed, the net result being a largely phoneme independent lip-sync. In the example embodiment of Fig. 4, each track of audio coming in has an associated SAM 402 and MCU 404. This permits separate analysis and classification for separate characters, so that multiple characters can be talking at the same time and the analysis for each can occur in parallel (i.e., the mouth states are generated separately for each individual character). This also allows for multiple characters to be simultaneously animated and lip synced. In other embodiments, where time is less of a factor, multiple audio inputs can share a common SAM 402 using known techniques such as multiplexing (Fig. 5A) or by serially analyzing audio tracks and accumulating the results before provision of an audio track to the MCU 404. In similar fashion, multiple SAM's 402 can share a common MCU 404 (Fig. 5B), although additional complexity or increased processing time may result. Of course, in the simplest case an arrangement like that of Fig. 4 can use a single audio track with a single SAM/MCU pair by making a series of passes. It should now be apparent that there is no required correspondence between the number of SAM's 402 and MCU's 404 for any particular embodiment and that the correspondence can vary while preserving the independent benefits or features each provides.

The SAM 402 takes a digitized audio input from the audio interface 302 or a stored file 400, converts it into a spectral representation using a Fast Fourier Transfer (FFT) technique.

Stated another way, the input to the SAM 402 is a signal in the time domain -- the audio signal -- and the output of the SAM 402 is the same signal represented in the frequency domain. The output from the SAM 402 typically consists of six values. Four of the values (f0, f1, f2, and f3) are associated with the power of the signal within defined bands (b_0, b_1, b_2 and b_3) defined by Bandwidth Settings 406 after taking the FFT. A single value is derived in each of the bands representing how much of the spectral content exists within that band. Although not necessary, for convenience, the values are normalized over the total power to a value between 0 and 1. In addition to the spectral content, a zero crossing value (z) is also output by the SAM 402. Zero crossings refer to the number of times that a signal will go above or below zero within a particular time window, and also gives some indication as to the noise level of the signal. The final value that comes out of the SAM 402 is the overall power (p) of the spectrum within the four bands. This gives a general sense of how loud the character is speaking at a given time and how much spectral power is being output at a given time. The SAM 402 doesn't usually use overall amplitude, just spectral power. One illustrative reason for using overall power is for a sound like an "S" sound, the amplitude may be very low on the signal, but there's quite a bit of high frequency spectral content in it because it's a very noisy signal.

The combination of the six values create a pattern that is presented to the MCU 404, which examines in the spectral content within bands, zero crossings and power values. When a pattern applied to the MCU 404 is matched, this indicates that a particular mouth shape should be used. The MCU 404 analyzes the six valves by comparing each value to see whether it falls within a prescribed range. The set of ranges correlates particular mouth shapes with the pattern defined by the six valves. One example of this correlation is shown in Table 1 below. In this manner, phoneme independence is largely and advantageously achieved.

At this point it is worthwhile to note that, for purposes of explanation, a set of mouth shapes introduced by Hanna-Barbara is used because it is a commonly used convention. It will be understood that the same technique of matching the FFT pattern with mouth shapes can be employed with any set of mouth shapes. According to Hanna-Barbara convention there are nine mouth shapes, labeled: A, B, C, D, E, F, L, V and M. In Table 1 (called a Classification Table 408), mouth shapes are correlated to the six values so that each row corresponds to a mouth shape (although some mouth shapes have several rows to account for different patterns or sound variations) and the columns are example minimum and maximum values for each one of the six values f0, f1, f2, f3, p and Z. The numbers at the intersection of each is the range within which each value must fall for each mouth shape.

TABLE 1

Mouth		f0		f1		f2		f3		p		z	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
M	Noise	0	10	0	1	0	1	0	1	0	1	0	1K
	Breath	0	50	0	10	0	5	0	1	1	5	1	60
A	B/P (#1)	0	0	0	0	0	0	0	0	1	2	0	600
	B/P (#2)	0	40	0	10	0	10	0	0	1	2	0	600
	"emm"	50	95	0	40	0	10	0	5	1	20	10	100
V	V	10	50	10	25	10	30	10	30	1	20	400	700
	F	60	93	4	50	2	40	0	90	1	6K	150	500
L	T	0	15	0	20	30	92	10	30	5	6K	80	450
	"nd"	0	15	0	20	30	92	0	10	5	6K	80	250
	L	25	85	0	16	0	10	50	75	2	2K	200	400
B	"ee"	10	97	0	35	1	85	0	15	1	12K	15	400
	"ess"	0	50	0	5	0	80	0	100	1	12K	200	650
F	"oo"	97	100	0	10	0	4	0	1	2	6K	15	120
	Filter	93	100	0	10	0	4	0	1	*	*	*	*
E	ER1	40	96	0	50	0	11	0	5	1	15K	15	110
	ER2	0	60	90	100	0	11	0	5	1	15K	30	90
D	hard1	10	80	10	80	0	20	0	5	15K	40K	30	150
	hard2	0	30	10	99	0	60	0	10	15K	40K	30	180
C	"eh"	10	80	10	80	0	20	0	5	1	15K	30	200
	"ah"	0	50	10	99	0	60	0	10	1	15K	30	225

* = Don't Care

This classification table 408 illustratively contains representative values. Depending upon the implementation, the values in the table may vary from speaker to speaker and may also vary from animation director to animation director. In sum, the classification table 408 merely provides a range of values that need to be matched in order to output a particular mouth state.

Moreover, although shown as a table, the same concept can be employed using an algorithm or formula, or using software such as shown in either of Figs. 10A through 10R or 10S through 10LL, or by implementing the arrangement using hardware created, for example, using an electronic design automation development tool, silicon compilers and/or by employing architectural synthesis techniques.

In another alternative variant, the classification can be done by converting the numerical values in the pattern to a symbol representing any value in a range. For example, particular ranges of values may be identified as zero(Z), very low (VL), low (L), medium low (ML), medium (M), medium high (MH), high (H), very high (VH) and maximum (X). Once classified into these ranges (i.e. a symbol has been assigned to the value), a search can be performed to identify the corresponding mouth state or mouth identifier for a particular set of symbols. For example, to identify an "A" mouth, the set of symbols for spectral content should be between M,Z,Z,Z and X,Z,Z,Z ,with the zero crossings being in the range of VL to VL and the overall power being in the range of VL to X. An example of this variant is illustrated in Figs. 10S to 10LL.

Fig. 6, shows a representative audio input signal 602 and Fig. 7 shows a sample output 702 of the FFT for an audio input signal. A well known artifact of FFT's is that FFT's can potentially introduce frequencies which are not present in the input signal. There are several known ways to mitigate those introduced frequencies. One technique involves using an

amplitude envelope over a time period. Multiple amplitude are employed and envelopes overlapped to eliminate or reduce undesirable clicking when the FFT goes from time period to time period.

The FFT time is a function of the audio signal clock and the number of samples in the FFT. Depending upon the specific application, an FFT sample size of 1024 or 512 is generally used, although other FFT sample sizes could be used, for example, if the number of bands were different or to allow for a different prefiltering scheme. With the 1024 sample FFT a sample clock of 44.100 kHz is used. The FFT time would then be 1/44.100 * 1024 or approximately 22 milliseconds. In the embodiment of Fig. 8, a 512 sample FFT is used. In both cases, the upper half of the FFT is discarded so, for the 1024 sample FFT, only the first 512 frequencies are analyzed in generating f0, f1, f2, and f3. Similarly, if a 512 sample FFT is used, only 256 frequencies would be used.

As noted above, there are alternative techniques that can be used for FFT prefiltering. Some examples are Hamming windows, square wave amplitude modulation and non-overlapping windows. In general, for most speech applications, the most desirable FFT uses overlapping triangular windows 604, 606. The duration that the FFT runs (t) is the width of a single triangle filter 604. The filter width (t) is also the inference period of time for the mouth state.

Fig. 7 shows the relationship between the spectrum returned by the FFT and the bandwidth settings 406 of Fig. 4. These bandwidth settings 406 are conventionally set to roughly approximate where a formant in speech would appear. This technique uses a narrow band and very detailed sampling of the low frequencies and, as you go up in the frequency spectrum, the bandwidth in each band becomes wider because there is less information present at

the higher frequencies. The power in each band is calculated and is summed to obtain the overall power (p).

The power in bands b_0, b_1, b_2 and b_3 704, 706, 708, 710 give an indication as to what type of sound is being spoken. A lot of very low frequencies in band b_0 with a little bit in b_1 indicates possibly a sound similar to an "R" sound; a very high value indicating a lot of power in b_0, but zero power in all the other bands indicates a sound similar to the double "O" sound as in "food." Conversely, low power in bands b_0, b_1 with a little bit of power in band b_2, and a lot of power in band b_3, usually indicate an "S" sound as in "Stephen" or "Z" sound as in "zone" or "xylophone."

Fig. 9A is a more detailed functional block diagram of the Mouth Classification Unit 404 (MCU) which actually applies the classification table 408 to the output from the SAM 402. The input to the MCU 404 is the power in the four bands, b_0, b_1, b_2, and b_3, the zero crossings (z), and the overall power (p) derived from the FFT. These values are then classified, for example, in the illustrative examples of Figs. 10A-10LL by number or symbol and, when patterns are found, the output is one of the different mouth states 902. The mouth shapes in the example are also labeled A, B, C, D, E, F, L, V and Z, according to the Hanna-Barbara convention. According to this convention, the "A" mouth is generally used for the "M, B" and "P" sounds (the closed mouth sounds). The "B" mouth is used typically for "S" sounds and "ee" sounds, and is usually drawn as the teeth being closed and the mouth being open. The "C" mouth and the "D" mouth are used for most of the consonants, the "C" mouth is used for the quieter consonants, the "D" sound is used for the louder consonants. The "E" mouth is used for an "R" sound and a soft "O" as in "Rob." The "F" sound is used for the "W" and, "oo" type sounds. The L mouth is used for the "L" sound as in "lung," or for the "tch" as in "touch", as well as any sound that would require

the articulation of the mouth to suggest movement of the tongue. The V mouth is used for "V" sounds and for "S" sounds. The M mouth is generally also considered the neutral mouth position and is used when no sound is being spoken. Fig. 9B is an example alternative variant having a slightly different pattern match search order.

The pattern matching 904 is done with a simple scheme that conditionally looks for values within ranges. This may be done in software such as in either of the examples of Figs. 10A-10LL, using, for example, MAX which is an object oriented programming environment for music and multimedia and is commercially available from Opcode Systems, Inc., 365 East Middlefield Road, Mountain View, CA 94043, or in hardware, using, for example, conventional digital logic elements or specialized application specific to integrated circuit (ASIC) or programmable logic type (FPLAs, PLAs, PLDs, etc.) devices. The pattern matching procedure is done by sequentially by typically looking for the V mouth patterns first, followed by the L mouth, S mouth, E mouth and B mouth, then the A mouth, and the M mouth patterns then follow. In other alternatives, the sequence can be V mouth followed by L mouth, followed by B, A, M, F and E mouths in sequence.

The C and D mouths patterns are covered last because they are used for sounds that are not specifically categorized, but generally indicated by overall spectral power, with low spectral power indicating a C mouth and higher spectral power indicating a D mouth. Depending upon the particular implementation, the C mouth can be searched for before the D mouth or vice versa. If a pattern match failure takes place for any reason after the above attempts, an inference can be made based upon zero crossings, or the combination of zero crossings and power.

To generalize the process, it is one of successively increasing the latitude for a pattern match. To select a V mouth a fairly precise pattern match is required, whereas to select the D mouth a fairly broad pattern will do.

In certain cases, the pattern match procedure order may vary to accommodate user preference or character idiosyncrasies. For example, in some cases, the B mouth is searched for before the F mouth whereas in other cases the reverse is true. In still other cases, a particular pattern, for example the V mouth pattern, can be disabled, dispensed with or varied slightly to customize or "tune" the correlation to a particular speaker or maintain character related consistency.

An added level of processing can be optionally added to reduce the amount of fast oscillation between mouth shape and mouth shape or "mouth fluttering," which is considered to be visually undesirable. In one approach, simple rules are applied by looking at mouth patterns over three different samples. For example, if there's an open mouth, a closed mouth and then an open mouth in succession, the second open mouth is discarded to keep the visual lip-sync looking smooth, not erratic. A visual analysis of simple characteristics can be used to identify the undesirable visual effects and filter them out by applying rules in a similar fashion.

While the general technique is to match patterns in order to sequentially identify mouth shapes, and thereby transition as necessitated by the audio input, for example from the V mouth to an E mouth, animation directors sometimes want to make the transition happen more slowly or and more smoothly, and introduce subtle gradations between the V mouth and E mouth or, in general, between any mouth and any other mouth. The MCU 404 is flexible enough to allow both an instantaneous change from mouth shape to mouth shape, or a transition from mouth

shape to mouth shape, as well as to allow for the filtering out, or introduction of, particular mouths for sounds based upon how it appears to the animation mouth director.

The SAM 402 and MCU 404 also differentiates this technique from the phoneme type of lip-syncing technique conventionally used which requires a complex analysis involving looking for stops, affricates, fricatives, nasals, glides, etc.. Although there is some broad correspondence between some mouth shapes and some phonemes, there are many situations where the spectral analysis technique is too broad or too coarse to be considered accurate phonetically. Advantageously, by using the spectral analysis and classification technique described herein, no particular attempt is made to find and parse words so there is no specific phonetic analysis that needs to be done. There are no phoneme tables involved, just an analysis of the audio signal and a heuristic association between a pattern of frequencies, zero crossings and power to a mouth shape.

Using the signal analysis and classification technique, there is no language dependency because no language elements are being looked at or examined for, phonetic structure - - rather the approach is simply a matter of matching a particular sound with a particular mouth shape. The "oo" sound, whether it's produced as part of a word in Arabic, or Japanese or English, will use the "S" mouth, which is the appropriate mouth for that particular sound. Thus, by using multiple SAM 402, MCU 404 pairs, multiple language versions of the same animation may be created, with each having its own proper lip sync.

In some embodiments, the technique is also, tunable. The bands f0, f1, f2 and f3 are tunable, in software or hardware, to account for the differences in speakers, languages or intonation, or to suit a particular animator's sense of aesthetics. Tuning also is done by changing

the values in the classification table 408, the bandwidth settings 406, and the heuristics used to filter out undesirable visual effects.

It should be noted that although the classification process performed by the AAU 100 and its SAM 402, MCU 404 components the same technique can easily be implemented in hardware which could, in some embodiments, significantly increase the speed or reduce the size of the computer hardware that is required to perform the mouth classification. A hardware implementation could however result in loss of some of the flexibility in the tuning.

Nevertheless, the classification tables 408 could also be modularized and programmed, for example, into a flash memory or EEPROM so that they could at least be tuned minimally or into a PROM or EPROM if tunability is not an issue. As will also be recognized, the SAM 402 can be implemented in hardware while the MCU 404 is done in software or vice versa.

For general speech, minor tuning seems to produce fairly good classifications which work well for a wide variety of voices. For example, men and women speaking into the same AAU 100 can produce reasonably good lip-sync without tuning the classification tables 408 or the bandwidth settings 406 in between.

The tunable nature also allows for professional quality animation for a very demanding animation director who wants specific directorial control over the movement of the mouth and/or lip-sync.

One of the further advantages of the instant approach is that the classification tables 408 can also be empirically generated by running a particular audio segment over and over again and making a visual correlation between the pattern of spectral content, power and zero crossings and the resulting mouth shapes. That classification of mouth sounds can then be further tried, over a

wide variety of words or phrases the character might be using, and iterated until the desired visual effect is achieved.

Similarly, using two, three and five bands for purposes of classification can also produce favorable results although, in general, two and three bands do not work as well as four bands and greater than about five bands introduce ambiguities that typically exceed acceptable levels. Moreover, it is to be appreciated that, in some cases, any two sequential bands may not be contiguous (i.e. there may be a gap between them). Additionally, the number of bands described herein refers to the number of bands used for classification purposes -- in other words, the number of bands used for purposes of measurement is may significantly exceed five bands, however in those cases, the values in measurement bands will be combined to create the two to about five classification bands. By way of example, band b_2 may contain values from a single measurement band or be consolidated from 2, 3 or even 1024 or more measurement bands.

Having described the various functional components and some exemplary implementation variants, the overall operation will now be discussed.

An animation is typically a series of scenes, where a scene is made up of a number of different components or elements which are layered together and composited to create an animation frame. Generally speaking, scenes are made up of a background element, one or more characters and possibly some foreground elements which may be in front of the character. Layering is done so that the background is at the back, the character is in front of the background and the foreground elements are on top of the character. There also may be situations where there are no backgrounds or the background is blue and the character sits on a blue background for compositing using a traditional chroma-key method. Compositing combines the pieces together into a series of animation frames.

Referring now to Fig. 11 which includes an example character. The example shows the character elements as will be assembled onto a background 1102. The character 1104 is illustratively made up of a number of elements: a head 1106, eyes 1108, mouth 1110, body 1112, left arm 1114 and right arm 1116. Each of these elements animates independently, or moves independently. As a result, the layering is such that the eyes 1108 are placed on top of the head 1106 and the head 1106 is placed beneath the mouth 1110, the mouth 1110 is sits atop the body 1112 and the arms 1114, 1116 will change their layering depending upon their gestures. The left arm 1114 sits on the left side of the body 1112, and the right arm 1116 lays on top of the body 1112 on the right side, and finally, the foreground element 1118 sits and covers a small portion of the bottom of the body 1112. In this example these elements constitute a scene. Each of these components may have animated gestures associated with them including changes to the background or foreground and all may be manipulated or changed independently. Thus, Fig. 11 shows by example the basic components and how they are arranged in relative priority or layer order. The character 1104, when it is composited together (i.e. with a particular body with particular head positions and arm positions etc. . .) or assembled from a particular point of view, is generally considered "posed." Poses can range from different camera angles to different body states and positions. Characters are organized into poses so that all the necessary elements can be created, assembled and used within a particular pose in any sequence. Fig. 12 shows four example poses that this character 1104 can assume. One pose has the character standing with his arms at his side and standing somewhat upright (Pose A). In Pose B, the character is leaning forward on the podium. In Pose C, the character is leaning forward more and in Pose D, the character has one elbow resting on the podium with his shoulder basically turned to the viewer. The way this particular character was designed, the head elements are all the same in all four

poses so the facial expression elements are compatible with all of the poses. The arms are the elements which incompatibly vary from pose to pose. In other words, the arms in pose A could not be used on pose D, C or B, but the head in pose A and its elements are used in pose B, C and D except, in Pose D, they are translated slightly down in the frame to correspond to the body moving forward. All the elements that were drawn for the head in pose A are reusable in the other poses. In contrast, there are unique animations and drawings for each of the arms in each different body pose. This organization of the possible poses is called a "character table". So, for this example, the character table shows the four poses and the relationship between the background character and foreground in a composite frame.

The list of legal or possible gestures defined for each of the different components within a given pose is called a "gesture table". Fig. 13 is an example gesture table 1302 for the character of Fig. 12 in Pose A. In this example gesture table 1302, there are five elements which can possibly change -- the left arm, right arm, head, mouth and eyes. The left arm is the element for which there are the largest number of defined gestures that the character can go to in that particular pose and each has a corresponding image. They are the arms at the character's side 1304, arms on the hip 1306, finger up in the air 1308, palm raised high 1310, the palm is low 1312 and another gesture where the character is reading from a card 1314. Although the images are not shown, in the gesture table 1302 the right arm has a defined corresponding set of atSide and onHip gestures. It doesn't have a corresponding fingerUp, but it does have a palmHigh. Similarly, the head has a profile left gesture, a ¾ left, a front, ¾ right and profile right gesture. The range of head gestures corresponds to the character turning its head from profile left through to profile right, so the character can move its head from left to right or vice versa. Attached to each of these heads are a set of mouth gestures, each containing the mouth shapes which will be

selected as a result of the audio analysis. The "up" and "down" mouths allow for lip-sync involving different facial expressions. A down turned set of mouths are usually used to indicate anger, sadness, etc . . . . Up turned mouths are for normal or generally more congenial expressions. This organizational arrangement allows for many different mouth expressions for the character, for example, overly big or wide mouths; mouths to the side; mouths which vary based upon loudness, and so on. In addition, the eyes have a range of gestures, for example, they can be wide eyes or half mast. Half mast eyes are often used to indicate a level of boredom or exhaustion.

The frames database 132 of Fig. 1 is, to some extent, an image "library" for each element which may change at any time during the animation. Of course, it may also contain images which do not change or are not used. In fact, the frames database 132 will likely contain more images for a particular element and action than would actually be used during a given transition, or even a given scene or animation. However, those images are advantageous if expandability or future use is a consideration. For example, if an animated series is contemplated and time is available, it may be desirable to fully define all of the possible actions for a given character. This allows extremely fast, low cost production of high quality additional episodes and specials.

Another advantage to fully defining a movement will be evident from the following example. Assume that a given element's change from Pose A - start position to Pose A - goal position is fully imaged using a linear change over 10 frames. The sequence portion of the transition table described below can be used to define that change using all 10 images or some subset thereof. In most cases, the sequence will be a subset of the images in the frames database 132. If all ten images are used, the motion will take 2/3 of a second to complete if shot on twos. If a quicker transition was desired, 5 images can be used, for example images 1, 4, 7, 9, and 10.

This would provide a faster transition movement with a smooth finish. Alternatively the same fast transition might involve using only images 1, 3, 5, 7 and 10 for a more continuous visual effect. Similarly, a slow to fast transition might use images 1, 2, 4, 7 and 10. Moreover, if a reverse movement needs to be defined, it can be done independent of the forward movement. For example, the forward movement might sequentially use images 1, 4, 7, 9, and 10 and the reverse use, images 10, 6, 3, 2, 1. Thus, a high degree of flexibility is afforded. In a similar vein, if during production, the 10, 6, 3, 2, 1 transition was deemed too slow and visually undesirable, the animation could be quickly changed merely by editing the values in the transition table to, for example, to use images 10, 5, 2, and 1. Fig. 14 illustrates the selected images from the frames database 132 used for an in-pose sequence 1402 and for a pose-to-pose sequence 1404.

As will be evident, the frames database 132 for this character and element contains at least 11 and more likely up to 103 images. The selected images (Images 1-7) are the ones which will be sequentially used during a transition of the arm within Pose A from atSide to onHip and from Pose A onHip to Pose B onDesk. Those image transitions are shown in what is referred to as a "sequence table" which is a table that takes the component from one gesture through to another. It is the sequence of transitions that the component would have to go to achieve a desired gesture from a given starting place. The sequence table can reflect any changes which will occur in the frames. So, for example, in addition to changes involving movement, the changes could involve a change in look, like a costume change, so that the same sequence is defined twice, once for a green jacket and once for a blue jacket. Each one of which would then be a separate entry in the gesture table 1302 and have a corresponding sequence.

There are two different kinds of sequences in this organizational arrangement: sequences which are transitions while staying in a particular pose and transitions that take you from one pose to another. Thus, Fig. 14 shows two parts of the sequence table. The sequence table of Fig. 14A shows the starting image, ending image and inbetweens for a transition between the atSide gesture and the onHip gesture. The transition is done with seven images, all taking place within Pose A.

The example shown in Fig. 14A is a sequence or a transition from the arm for character named "Buddy" in a two shot for pose A (standing upright). This sequence takes the left arm from the at Side gesture to the onHips gesture. In this example, seven images are used to make the transition from atSide of onHip. The first image is black, because the body of the character hides that arm in that particular image. In image two the arm starts to become visible. By image three the arm has a very visible appearance. By image four the hand clenches as a prelude to resting on his hip. In image five, the arm comes fully up and then finally, in image 6 it begins to settle and in image seven ends in its resting place. This sequence of images, it will be recognized, uses the conventional animated effect of cushioning - - for the settling of the arm into its final resting place to give the arm a natural appearance. So the seven images are used to make the transition from "at side" to "on hip."

The sequence table of Fig. 14B shows the pose-to-pose transition between the last image in the in pose transition (Image 7) where the arm is in the onHip and it is making a transition from that pose to the onDesk arm pose of Pose B. The transition from Pose A left arm onHip to the pose B left arm onDesk is effected in four frames, starting with the onHip and using two inbetweens going to the onDesk pose. During this transition there also is at least a corresponding

new body transition with in between steps, however those transitions would be accomplished in the same manner and, accordingly, are not shown.

A transition table 131 is the overall structure that's used to coordinate the animation and movement of the character from state to state. In other words, the transition table 131 is the "glue" that holds the whole character together and its how the animated movement from state to state is described and constrained. Fig. 15 is an illustration of a small part 1502 of a transition table for the whole character. The part shown is for the character in pose A, and its assumed that the left arm is currently at the side. The transition table describes the only legal transitions and, in particular, how the transition from atSide to stayatSide, or atSide to go to: onHip, fingerUp, or palmHigh. The list of in Pose transitions in this Fig. corresponds to the gesture table described above. The second entry is the list of in Pose transitions including from pose A left arm atSide to pose A left arm onHip which is the sequence table of Fig. 14A. The values in the cells of the table show the frame numbers that are required to make that transition. The second section of the transition table describes the pose to pose transitions. In its simplest embodiment, the individual entries in the transition table each represent individual steps. In some embodiments however, boxes between two steps may be left empty or undefined for a particular aspect. In that case, the lack of one or more intervening values could signal the need to interpolate values between those two points, using for example, known linear interpolation formulae or morphing techniques. In those embodiments, as the transition controller 128 encounters empty boxes, a routine is invoked which "looks ahead" to determine what the goal state is, advances to the first step for which there is a value and "looks back", or maintains a count of the number of steps between the current state and the goal state. For example, if there is an initial value in step no. 1 and the goal is in step no. 7, and only step nos. 2 and 3 have values, an interpolation would be

done for the steps between step 3 and step 7. Further rules could be imposed to require the interpolation to take into account the number of steps in between, or apply some other rule or constraint consistent with the desired effects. The pose to pose transitions and a number of different possible in pose transitions are shown in Fig. 15 for the example illustrated in Fig. 14B. The pose to pose transition illustrated is a Pose A to Pose B transition for the left arm, with the arm gestures for the arm being: atSide, onDesk, tapDesk, and pointWest. For simplicity, only the values for the transition from Pose A left arm atSide to Pose B left arm onDesk are shown.

The transition tables are used by the transition controller 128 when trying to bring the character from the current state into a desired or goal state. A trigger is presented to the transition controller 128 and the trigger indicates that the character's current state is atSide and desired transition is that the character move its left arm from atSide to onHip. The transition controller 128 detects that a new goal state is being presented and checks the current state of each element of the character to determine whether it is in the goal state. If it is not in the goal state, then it causes the next step to be taken to get to that goal state. This iterative process of querying each of the components continues until all elements report that they are in the desired state.

The transition table may also include different classes of default transitions which are used in the event that an exact match does not exist. This can occur, for example, if the transition calls for a move from Pose A are atSide to Pose D arm atChin and that transition is not defined directly. In that case, the Pose-to-Pose transition may involve a Pose A arm atSide to a "neutral" position defined as a default with the likelihood that a transition from that neutral position to Pose D arm atChin exists. Thus, if the defined transitions fail, a check is made to see whether, or not, the desired state falls into the Class I candidates. If a match of one of the

candidates in Class I is found, then that's the transition it uses. If none of the classes are matched, a final default may be defined. For example, for one character the default may be to put their arm down when they are trying to go to a state which is not a legal state. For other characters, the default might just leave the arm in a particular state. An example of when this would be used is where a body suit on an actor moves in a way that is not defined in the gesture table. Another situation where a default would be used is to allow characters to evolve over time. A particular transition from a previous to a new state may not have been defined because of cost or budget constraints, the complexity of drawing it, etc. . . . In order to keep the character from falling apart, a neutral default action is defined to allow for the addition of further transitions from that neutral state at a later time. If pose A has a left arm atSide, pose B has a left arm atSide, pose A has a left arm onHip but Pose B does not have a pose arm onHip. If the character has its left arm on its hip and it is coming to a pose B position there is no corresponding onHip for Pose B. The default would be used to bring the arm down to the side, which it is a legal pose for Pose B. In that way the character can make the desired transition in a coherent and visually acceptable way.

These transition tables can be extensive. They will generally be as extensive as necessary to create the desired fluidness and flexibility in character movement. Every component in every state, and/or every component in every Pose, has a list of legal gestures that can be made from that state or from the existing pose to any other pose. The default and classification methodology also allows the table to be asymmetric. The organization of the transition tables also allows the transition table to be dynamically edited and modified as necessary and allows the animation process to proceed smoothly and systematically. Taken together, the frames database 132 and the transition table allows for character consistent, truly "live" appearances by

the character, via an actor with a motion capture suit, or puppeteers manning a series of controllers, without the risk that the character will move uncharacteristically or in a manner contrary to the animator's wishes, irrespective of who is controlling the character. Additionally, if the character has particular vocal characteristics, those can be preserved for the "live" performance as well, for example, through straightforward adaptation, without the character suit, of the system and technique described in U.S. Patent No. 5,327,521 incorporated herein by reference.

The frames database 132 also allows for multiple versions of the same animation to be simultaneously created. For example, by using three different frames databases, one where a particular character is male, one where a particular character is female, and another where the character is an anthropomorphic animal. By feeding the same compositing commands to either three compositing engines, each having one of the frames databases or by using the same compositing engine with each frames database in turn, three different animations are created. This can be extremely advantageous for targeting of diverse markets where, for example, it is unacceptable to portray females in particular jobs in one cultural market, but highly desirable in another. Both markets can now be satisfied with a minimum of extra work.

Another use for the frames database 132 in conjunction with the timeline recorder 134 is to permit off-line rendering of a real time performance so that it is compatible with high resolution film or HDTV. In this situation there would be two (or more) frames data bases 132 one for the real time performance at video resolution and one for the final output medium at high resolution. Once a performance is captured, and possibly edited, in the timeline data recorded by the timeline recorder 134 using the video resolution frames database, the final timeline would be

played back using the high-resolution frames database and the composting engine would output high resolution frames to the high resolution film recorder or HDTV recorder.

As shown in Fig. 16, the entries that are stored in the table cells can also be a variety of different types of data items. They can be single integers (Fig. 16a) or frame identifiers (Fig. 16b) which are used to select a particular frame number out of the frame data base 132, and which indicate the image that should be used for that component for that step of the process for that transition. Other entries that could be stored include X and Y position on the screen (Fig. 16c), like the head position will be the same frame number, but would have a different X, Y location in Pose B versus Pose C versus Pose D. The entry can be a change in position (Fig. 16d). The entry can also specify orientation in degrees of rotation in the plane of the image, the entries can be X, Y and Z values, or it could be location in space or orientation using spherical coordinates (Fig. 16e, 16f) rather than Cartesian coordinates. The transition table can also contain "tag" values (Fig. 16g) so that items can be requested by name and be application dependent, etc. . . . Similarly the entries could represent change or Δ of position, angle, etc., a scale, zoom or magnification value (Fig. 16i), a factor that would allow the character to change its size, for example, while the character makes the transition from farther away from the viewer toward the viewer, causing a perspective change in size, and/or an indicator of extra long (ELS), medium long (MLS), long (LS), medium (MS), close-up(CS), point-of-view (POV), wide angle (W), soft focus, selective focus and tilted shots. Table cells can also contain scalar values augmented by geometric transformation, for example, rotation, scale and position, or parameters for properties such as color, shading, lighting, etc. The entries can reflect camera effects such as panning, tracking, fade, dissolve, superimpose, wipe, and slow or accelerated motion.

Entries can also be any kind of information used to control other applications, so you can make requests outside of the domain of the current application, and control other applications with it, for example, a motion capture 3D animation system providing it has program inputs. The control information can be a pointer to a program segment (Fig. 16k) or program segment itself stored in the cell of the table (Fig. 16l) which used to control the system or affect some external application. This is useful in motion capture in which some kind of rig or suit that captures the actual performance of a human being and their location in space. While motion capture works very well if you want to replicate human movement, it does not necessarily work well when the motion capture system is used to control animated character movement since animated character movement is not necessarily human-like in its transition. Depending upon the application and contents of the transition table, a whole control system is created which uses a wholly different animation methodology and which still operates in real time, but allows the coherence of the animation to be specified by an animator, rather than the dancer or performer. This allows the character to remain true to its conventional animation and characteristic behavior, even though it is performed live and somewhat independent of the specific performer controlling it. In the Idolon/Scrim embodiment, the transition table organization allows the cells to include as entries, in addition to the contents described above, information pertaining to Idolon placement, as well as factors such as warping, degree of transparency or opaqueness, deflection or twist, or even thickness if desired. With this embodiment, the composite frame produced by the compositing engine is made up of these Idolons with the images hanging on them positioned in space one behind the other to create a layered compositing effect. This embodiment uses a real-time 3D rendering system, such as openGL or the like.

Additional Alternatives

It will now be apparent that the techniques and principles described herein are, to some extent, a powerful hybridization of general animation techniques, such as frame compositing using a computer and conventional puppetry, where a character moves under external control. To that extent, the technique may be considered a form of animation under puppeteer (human or automated) control referred to as "animateering", "animatry" or "cartooneering" depending upon the particular context.

There are also a number of applications of the principles of the invention which would not be considered by some to fall within the category of classical animation. Nevertheless, the above principles may be straightforwardly applied in those applications in the following ways.

It should be evident that the above principles can easily be applied to cutout animation. To do so, cut-outs of the individual elements are made and imaged in as many different positions as is necessary. Each image would then be stored in the frames database for use as described above. Similarly, cutout elements can be placed on Idolons for manipulation.

An offshoot or variant of cutout animation uses a set of photographic images which may be manipulated in a similar fashion. In the most basic application, the source for the individual manipulable elements will be photographs (conventional or from a digital camera) taken in each possible pose through a series of transitions. However, for other than the most basic scheme, the process of creating the image database in this manner can be very time consuming. An alternative is to film the actor going through the range of motions and extract the elements from each frame of the film or interval of videotape. The advantage of this approach is that, not only is the line blurred between conventional motion picture and animation, but new films can be created from old movies and movies can be made using actors who are no longer alive in a form

of "anImageering". Moreover, to the extent films are recorded digitally, not only can component elements of body parts be extracted for use in a frames database, the creation of a library of mouths from a given actor can allow for foreign language dubbing with more visually acceptable results than achieved with conventional dubbing. To accomplish this result, mouth images are extracted from other frames of the film and are stored in the frames database since they are the images of the film which will change. The rest of each film frame is then treated as a background. The new audio track is then applied to the AAU 100 with the cells of the transition tables containing the information required to properly place the mouth images.

Depending upon the particular case, the extracting can be accomplished in a variety of ways, the simplest example being using masking techniques whereby undesired components of the frame are eliminated or blocked so that only the desired component is visible. For example, a sequence of frames showing a person swinging a golf club can be masked so that only one arm transitioning through the swing is visible in each frame. Having masked out the remaining components, the arm becomes a usable image asset that, by translating its position in space, can be used as an uppercut punch in a fight.

Another alternative is to use a rig made up of a number of cameras located at different points about the human character, for example, as if the person or some part of the person were contained within a fixed enclosure like a sphere, with the cameras being mounted on the surface of the enclosure and pointed towards the person. By moving through a range of motions, a series of images, from a number of angles, can simultaneously be captured for use in a frames database. A more limited version would concentrate on the head or mouth, to facilitate facial animation or lip-sync and allow for faster, cheaper and/or more accurate editing and/or dubbing.

Another application is a hybrid clay animation technique in which the individual clay elements are formed in the desired shapes and imaged. Again, those images, as individual components, can be processed as described above in a form of "claytoonery". Similarly, true puppet components can be imaged and stored in an image database for similar usage.

Notably, all of the above applications of some of the principles described herein have the same basic advantage -- namely the ability to allow third party manipulation of image elements within constraints imposed by the animator or director so that visual continuity and acceptability is maintained.

Other notable variants of system elements and methodologies using the same principles will also be apparent. For example, although the discussion has largely involved a tabular organization, it will be recognized that other organizational arrangements will work to a greater or lesser extent, depending upon the particular application. For example, instead of using tables or lists (which are merely single row tables), it is well known that formulas can be used instead. For example, where every third image in a set of images will be used, the notation "$i+3 \leq 10$" can be used instead of "1", "4", "7", and "10" in four cells of a table.

In another alternative embodiment, the transition tables can be used for pure in-pose transitions, with the pose to pose transitions from the sequence tables being handled using "procedurals". Procedurals can be used to bind together key poses in more elaborate animations than might be implemented using sequence tables. Procedurals reduce the complexity of the gesture database by combining the effect of many sequence tables into a single sequence. For example, they can be used to effect dramatic entrances and exits from a scene or they can implement gags and effects that change the entire appearance of the character at one time, they can also be used to turn over manipulable aspects of the character from external control to a

preprogrammed sequence and then back to external control. Alternatively, a procedural can have a preamble, defined by a transition sequence, to bring the character into a neutral pose that matches the start of the procedural. Once the preamble has completed, the sequence of frames that make up the procedural are run. After the procedural has completed, control can be transferred back or a postamble may be used to return the character to a specific neutral pose, again through the use of a sequence table. Procedurals can also be used in combination with sequence and/or transition tables to allow, certain elements to be externally controlled, while other elements have control periodically (or on a one-time basis) taken away and then returned so that, for example, a character can walk while waving its hands and talking with extreme facial expressions. All under external real time control. A procedural can be used to remove control of the arms and legs and run a sequence where the character trips, brushes themselves off and gets up -- at which point control of the arms and legs would be returned to the external manipulator. At all times however, the facial expressions may have remained under external control.

In other embodiments, logical components of the system can be modularized (whether in hardware, software, firmware, microcode or some combination thereof) for use in related applications, for example by employing a silicon compiler, design automation tool or using architectural synthesis techniques so that some or all of a functional component or two or more functional components are implemented in one or more chips. For example, the AAU 100 can be modularized for use in a children's toy, such as a stuffed animal, which is set up to form particular mouth shapes. A prerecorded audio input 400 would allow the toy to convincingly sing songs. Similarly, a microphone would allow a person to serve as the voice in real-time.

In still other embodiments, the logical components, whether implemented in software, hardware or some combination, can be separately packaged for integration with applications or

systems of third parties, for example, by creating a plug-in expansion card, a software extension or plug-in for an existing computer program or some combination thereof.

Figs. 17 through 75 are a program listing of one embodiment of a CE implemented in software suitable for use as described herein, as an alternative to a hardware implementation of a compositing engine.

Figs. 76 through 88 are an example software embodiment for each of the AAU 100 and TCU 128. This example is written in MAX, which is available from Opcode Systems, Inc., 365 East Middlefield Road, Mountain View, CA 94043 and additionally uses MSP, which is a set of object extensions to Opcode's MAX that synthesize, process, analyze and delay audio signals in realtime, available from Cycling '74, 1186 Folsom, San Francisco, CA 94103. Fig. 76 is the main program level in the hierarchy. There are several "patches" 762, 764, 766, 768 shown, each of which corresponds to a particular character. For example, one of the patches 763 is for the character "Buddy" illustratively used for purposes of explanation in Figs. 11 through 15. Fig. 77 is the voice portion of the program, which roughly implements one example software embodiment of the of the AAU 100 and MCU 404 functionality. Fig. 78 is the audio portion of the program which is a subcomponent of the voice portion of Fig. 77 and specifically implements one example software embodiment of the the SAM 402 and MCU 404 functionality. Depending upon the particular implementation, as noted above equivalents to the example software embodiment of the AAU and/or SAM (or sub-components thereof) can also be embodied in hardware.

Figs. 79 through 88 roughly implements one example software embodiment of the TCU 128 functions for the different components of the character of Figs. 11 through 15. As noted above, equivalents to this example implementation (or sub-components thereof) can also be

embodied in hardware. Fig. 79 is the highest level of the character hierarchy for a character, there would be one of these for each character patch which, at the lower levels would define those character elements subject to transitions. Figs. 80 through 88 implement the changeable elements for the character of Figs. 11-15. Specifically, Fig. 80 relates to head movement, Figs. 81 and 82 relate to eye movement, Figs. 83 and 84 relate to body movement, Figs. 85 and 86 relate to mouth movement, and Figs. 87 and 88 relate to arm movements. Of course, the program can be expanded in a straightforward manner for additional character elements (i.e. wings, ears, tail, etc.) or other elements of a frame subject to change, such as foreground elements, background elements, or parameters such as lighting, colors, opaqueness, twist, warp, etc. (i.e. those defined in the cells of the transition table).

Fig. 89 is an example procedural, written using MAX, which causes a character to react to pain with an "Ouch!".

It should be understood that the above description is only representative of illustrative embodiments. For the convenience of the reader, the above description has focused on a representative sample of all possible embodiments, a sample that teaches the principles of the invention. The description has not attempted to exhaustively enumerate all possible variations. That alternate embodiments may not have been presented for a specific portion of the invention, or that further undescribed alternate embodiments may be available for a portion, is not to be considered a disclaimer of those alternate embodiments. One of ordinary skill will appreciate that many of those undescribed embodiments incorporate the same principles of the invention and others are equivalent.